

# **smwrGraphs—An R Package for Graphing Hydrologic Data, Version 1.1.2**

Open-File Report 2016–1188  
Supersedes USGS Open-File Report 2015–1202



# **smwrGraphs—An R Package for Graphing Hydrologic Data, Version 1.1.2**

By David L. Lorenz and Aliasha L. Diekoff

Open-File Report 2016–1188  
Supersedes USGS Open-File Report 2015–1202

**U.S. Department of the Interior**  
**U.S. Geological Survey**

**U.S. Department of the Interior**  
SALLY JEWELL, Secretary

**U.S. Geological Survey**  
Suzette M. Kimball, Director

U.S. Geological Survey, Reston, Virginia: 2017

For more information on the USGS—the Federal source for science about the Earth, its natural and living resources, natural hazards, and the environment—visit <https://www.usgs.gov> or call 1–888–ASK–USGS.

For an overview of USGS information products, including maps, imagery, and publications, visit <https://www.usgs.gov/pubprod/>.

Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

Although this information product, for the most part, is in the public domain, it also may contain copyrighted materials as noted in the text. Permission to reproduce copyrighted items must be secured from the copyright owner.

Suggested citation:

Lorenz, D.L., and Diekoff, A.L., 2017, smwrGraphs—An R package for graphing hydrologic data, version 1.1.2: U.S. Geological Survey Open-File Report 2016–1188, 17 p., <https://doi.org/10.3133/ofr20161188>. [Supersedes USGS Open-File Report 2015–1202.]

ISSN 2331-1258 (online)

## Contents

Abstract .....	1
Introduction .....	1
Description of smwrGraphs .....	1
Graph Set-Up Functions .....	2
Main Plotting Functions .....	2
Components of a Finished Figure .....	3
Functions that Add to an Existing Graph.....	4
User-Support Functions .....	4
Internal Support Functions.....	5
Creating a Figure for Publication .....	6
Programmer’s Guide.....	7
Device Set-Up Function .....	7
Main Plotting Function .....	8
Function to Add to a Graph .....	12
Summary.....	13
Disclaimer .....	13
Acknowledgments.....	13
References Cited.....	14
Appendixes 1–9.....	15
Appendix 1. R Documentation .....	16
Appendix 2. Graph Setup Vignette .....	16
Appendix 3. Graph Additions Vignette.....	16
Appendix 4. Date Axis Formats Vignette.....	16
Appendix 5. Graph Gallery Vignette.....	16
Appendix 6. Boxplot Vignette.....	16
Appendix 7. Line and Scatter Vignette .....	16
Appendix 8. Piper Plot Vignette .....	17
Appendix 9. Probability Plots Vignette .....	17

## Figures

1. Graph showing components of a simple figure.....	3
2. R code for a specific example of a combination grap.....	6

## Tables

1. Functions in the smwrGraphs package that set up the graphics environment .....	2
2. Main plotting functions in the smwrGraphs package .....	2
3. Functions in the smwrGraphs package that add information or a plot to an existing graph.....	4
4. User-support functions in the smwrGraphs package .....	4
5. Internal support functions in the smwrGraphs package .....	5



# smwrGraphs—An R Package for Graphing Hydrologic Data, Version 1.1.2

By David L. Lorenz and Aliesha L. Diekoff

## Abstract

This report describes an R package called **smwrGraphs**, which consists of a collection of graphing functions for hydrologic data within R, a programming language and software environment for statistical computing. The functions in the package have been developed by the U.S. Geological Survey to create high-quality graphs for publication or presentation of hydrologic data that meet U.S. Geological Survey graphics guidelines.

## Introduction

The U.S. Geological Survey (USGS) uses graphics to present and describe hydrologic data in publications, presentations, and other venues. In the book “Statistical Methods in Water Resources,” Helsel and Hirsch (2002) discuss the importance of graphics for analyzing and describing hydrologic data. The purpose of this report is to describe a USGS created R package called **smwrGraphs**, which consists of a collection of graphing functions for hydrologic data within R, a programming language and software environment for statistical computing. The functions in the package have been developed by the USGS to create high-quality graphs for publication or presentation of hydrologic data that meet USGS graphics guidelines.

It is assumed that the reader of this report has a rudimentary knowledge of the R language and environment. General information on R can be obtained from the R project at <https://www.r-project.org/> (particularly the Manuals page). Many books and online resources also provide information about using R.

## Description of smwrGraphs

The R package **smwrGraphs** consists of a collection of functions to set up, create, and augment graphs within R. Functions in the package allow users to easily create simple, high-quality graphs and manage the data plotted to easily create descriptions of the plotted data. Additional functions are provided to add additional plots or annotation. Documentation for the functions is provided in appendix 1.

The functions in **smwrGraphs** are provided as a package in R (<https://www.r-project.org/>), an open source language and environment for statistical computing and graphics that runs on a variety of operating systems including UNIX®, Linux®, Windows®, and Mac OS®. R can be extended for additional functionality using packages. Additional information on the installation and administration of R and packages that extend R is available in the manual “R Installation and Administration” (R Development Core Team, 2011).

Many of the functions in the **smwrGraphs** package, version 1.1.0, have been ported from the USGS library for S+, a commercial statistical and graphical analysis software package (Lorenz and others, 2011). Other functions have been developed and used within the USGS since the publication of that report. All of the functions are in the public domain (see the “Disclaimer” section). In the text of this report, functions are given in *bold italics*, and arguments are given in *italics*.

The suggested citation for data from the **smwrGraphs** package can be acquired by using the *citation* function in R. The call is `citation(package="smwrGraphs")`.

## Graph Set-Up Functions

The graphics environment used by graphing functions in **smwrGraphs** must be set up by a call to the *setPage* function (for complete onscreen graphics device set up), *setGD* function (for a simple onscreen graphics device), *setPDF* function (for Portable Document Format [PDF] output), *setPNG* function (for Portable Network Graphics [PNG] output), *setSweave* function (within a SWeave script), or *setKnitr* function (within a knitr script). A call to one of those set-up functions can be followed by a call to the *setLayout* function to set up the layout for one or more graphs on the device. A call to the *setGraph* function is required to set up a specific graph area if *setLayout* is used. A complete list of graph set-up functions is provided in table 1.

**Table 1.** Functions in the smwrGraphs package that set up the graphics environment.

[PDF, Portable Document Format; PNG, Portable Network Graphics]

Function name	Description
setGD	Set up an onscreen graphics device.
setGraph	Set up the graph area for a single graph.
setKnitr	Set up a PDF file as the graphics device within a knitr script.
setLayout	Set up the layout for one or more graphs and optional explanation.
setPage	Set up an onscreen graphics device.
setPDF	Set up a PDF file as the graphics device.
setPNG	Set up a PNG file as the graphics device.
setSplom	Set up the layout for a scatter plot matrix.
setSweave	Set up a PDF file as the graphics device within a SWeave script.
preSurface	Set up a surface graph.

## Main Plotting Functions

The main plotting functions create graphs of data. All functions have options for controlling what is plotted and for controlling the axes. Table 2 is a list of these functions.

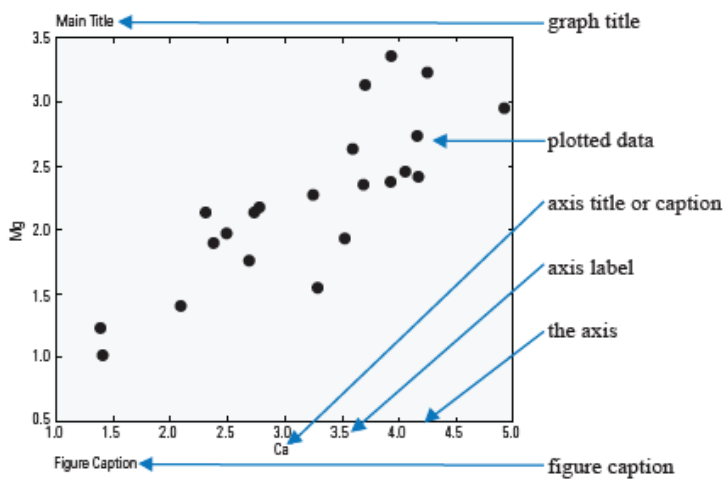
**Table 2.** Main plotting functions in the smwrGraphs package.

Function name	Description
areaPlot	Graph that highlights the area between curves.
biPlot	Graph two pairs of data.
boxPlot	Create a boxplot.
colorPlot	Create a scatter plot where the color of each symbol is set by another variable.
condition	Facilitate producing a series of graphs conditioned by a grouping variable.
contourPlot	Contour the data.
corGram	Create a correlogram from irregularly spaced data.
dotPlot	Create a scatter plot where the y-axis is a categorical variable.
ecdfPlot	Create a graph of empirical distribution function of data.
histGram	Create a histogram.
piperPlot	Create Piper diagram.
probPlot	Create a probability plot.
qqPlot	Create a Q-Q plot or Q-normal plot.
reportGraph	Create a report of any R object in a graph.
scalePlot	Create a graph with a fixed relation between the x- and y-axes.
seasonPlot	Graph time-series data over a year rather than by time.
seriesPlot	Graph seasonal data by season.
splomPlot	Create a scatter plot matrix.
surfacePlot	Create a surface graph.
stiffPlot	Create a Stiff diagram.
ternaryPlot	Create a ternary plot, also called trilinear or triangular plot.
timePlot	Graph time-series data by time.
transPlot	Create a scatter plot using arbitrary axes transforms.
xyPlot	Create a line/scatter plot.



## Components of a Finished Figure

The finished figure can contain a single graph or multiple graphs. The example in figure 1 identifies the components of a figure showing a single graph. The arguments in the main plotting functions that affect the drawing of each component or functions that add to a graph are described in the following paragraphs.



**Figure 1.** Components of a simple figure.

The graph area in figure 1 contains the components of a simple graph, including the figure caption. The graph area is defined using the graph set-up functions, such as *setPDF* and *setLayout*.

The plotted data are defined by the *x*, *y*, and *Plot* arguments. The *x* and *y* arguments define the data to be plotted, and *Plot* defines how they are to be plotted, as points or lines, the color, and so forth.

The y-axis is defined by the *yaxis.log*, *yaxis.rev*, and *yaxis.range* arguments, with similar names for the x-axis. Not all main plotting functions support all of those arguments. The *yaxis.log* and *yaxis.rev* arguments require logical values: TRUE sets up the axis in log-scale or decreasing scale (reversed), respectively, whereas FALSE sets up a linear axis

and increasing scale, respectively. The *yaxis.range* argument defines the exact range for the axis, which can be used to set the same range for multiple graphs or fine tune the automatic computation of the range. The default, *c(NA, NA)*, computes the range of the data for each axis, extends the range by a small amount to prevent plotting points on the axis, and then computes “pretty” limits (chosen so that they are 1, 2, or 5 times a power of 10). The actual axis length is controlled by the size of the graph area and the *margin* argument. The default value for *margin* is *c(NA, NA, NA, NA)*, which maximizes axis length while accounting for necessary axis labels and titles. In general, the default margin value should not need to be changed, except when the *setLayout* function is called to set up shared axes, or when right axis labels are needed.

The axis labels are defined by the *ylabels* and *xlabels* arguments for the y-axis and x-axis, respectively. In general, the default value is “Auto,” which inserts a reasonable number of labels given the range of the axis. An integer value also can be supplied, which inserts that number of labels for the axis, but the number of labels inserted is only a guide and may need adjustment to get the actual desired result. The *ylabels* or *xlabels* argument also can be a list that can help set up the axis and labels; if a list, then the component names must match the arguments for the *setAxis* function.

The y-axis or x-axis title, referred to as the axis caption in the USGS publications guide for illustrations (U.S. Geological Survey, written commun., October 2015), is defined by the *ytitle* or *xtitle* argument, respectively. The axis title is a single character string. For most functions, it defaults to the name supplied to the *y* and *x* arguments.

The graph title, called the graph heading in the USGS publications guide for illustrations, is created by the *addTitle* function. The graph title is included here for completeness. In general, the graph title is used as a short identifier for the graph, rather than as a description of the graph. In many cases, it can be a single letter for a multipart figure.

The figure caption is a detailed description of the figure, not just a graph. The figure caption is defined by the *caption* argument in the main plotting functions but also can be added using the *addCaption* function.

## Functions that Add to an Existing Graph

Once a graph has been created, several functions can add features to that graph. These functions are listed in table 3.

**Table 3.** Functions in the smwrGraphs package that add information or a plot to an existing graph.

Function name	Description
addAnnotation	Add text to a plot to annotate a feature.
addArea	Add a shaded polygon to the graph.
addAxisLabels	Label an axis.
addBars	Add a bar chart to the graph.
addCaption	Write a figure caption below the graph.
addCI	Add confidence interval lines to a simple linear regression or Q-normal graph.
addErrorBars	Add upper-lower error bars to data in a graph.
addExplanation	Create or add an explanation, also called key or legend.
addGrid	Add grid lines to a graph.
addLabel	Add text in the margin for specialized axis labels.
addMinorTicks	Add minor ticks to a graph.
addPiper	Add to a Piper diagram.
addSLR	Add a simple linear regression line to a scatter plot.
addSmooth	Smooth data and add the line to the graph.
addStiff	Add a Stiff diagram to a graph.
addTable	Add a small table to a graph.
addTernary	Add to a ternary plot.
addTitle	Add a title above the graph.
addXY	Add a line or scatter plot to a graph.
labelPoints	Label individual points on a graph.
refLine	Add a horizontal, vertical, or simple linear regression line to a graph.

## User-Support Functions

The **smwrGraphs** package contains some functions that can assist the user in preparing graphs. User-support functions are listed in table 4. The functions ending in “.colors” (such as *blueRed.colors*) can be used to help the user define a set of colors for points, lines, or areas to distinguish different sets of data. The functions *cov2Ellipse*, *dataEllipse*, *hull*, *interpLine*, and *paraSpline* provide specialized data manipulation functions to assist the user in adding specialized plots to a graph.

**Table 4.** User-support functions in the smwrGraphs package.

Function name	Description
blueRed.colors	Generate a set of colors ranging from blue to red.
coolWarm.colors	Generate a set of colors ranging from cool to warm.
copyDemo	Copy a demo script to the current or other directory.
cov2Ellipse	Construct an ellipse from a covariance matrix
dataEllipse	Construct an ellipse from x- and y-coordinate data
greenRed.colors	Generate a set of colors ranging from green to red.
hull	Get the data points for a convex hull.
interpLine	Interpolate x-values from y-values or y-values from x-values from plotted data.
paraSpline	Interpolate data using a parametric spline.
pastel.colors	Generate a range of colors in pastel shades.
redBlue.colors	Generate a set of colors ranging from red to blue.
redGreen.colors	Generate a set of colors ranging from red to green.
strip.blanks	Strip leading and trailing blanks from a character string.
warmCool.colors	Generate a set of colors ranging from warm to cool.

## Internal Support Functions

The **smwrGraphs** package has many internal support functions that are used within the main plotting functions and the other functions that add to a graph. These functions are listed in table 5.

**Table 5.** Internal support functions in the smwrGraphs package.

Function name	Description
boxPlotStats	Compute the statistics for a boxplot.
datePretty	Prepare the ticks and labels for a date/time axis.
frameWt, lineWt, stdWt	Set the line weight.
getDist.fcn	Get a distribution function for probPlot.
linearPretty	Prepare the ticks and labels for a linear axis.
logPretty	Prepare the ticks and labels for a logarithmic axis.
namePretty	Prepare the ticks and labels for a categorical axis.
numericData	Convert any data to numeric values
piperSubplot	Create projected rhomboid plot in the Piper plot.
probPretty	Prepare the ticks and labels for a probability axis.
renderBoxPlot	Draw boxplots.
renderBXP	The workhorse behind the boxplot.
renderX, renderY	Draw axis ticks and labels.
setAxis	Set up an axis from data.
setColor	Convert anything to a valid color.
setDefaults	Set defaults for plot control lists.
setExplan	Set up information for the explanation based on the plot control lists.
setGroupPlot	Set up information when the symbology is specified by group.
setMargin	Set margin for a graph.
setMultiPlot	Set up information when the symbology is specified for each point.
setPlot	Set the default value for a plot control list
ternarySubplot	Create ternary plot in the Piper plot.
ticks.render	Draw ticks on an axis.
timePretty	Prepare the ticks and labels for a time-series axis.
transData	Transform data to correspond to a specific axis type.
transPretty	Prepare the ticks and labels for an arbitrary transformed axis.

The **smwrGraphs** package has detailed help files for each function that may be accessed in the same manner as help for other R functions. Help features within R are further described in the manual “An Introduction to R” (Venables and others, 2011). The documentation for the **smwrGraphs** package is included in appendix 1 of this document. The **smwrGraphs** package also has vignettes (appendixes 2–9) that demonstrate the basic plotting capabilities and provide demo scripts that can be copied and can serve as starting scripts for more complicated graphs.

## Creating a Figure for Publication

The code in figure 2 provides an example of a combination graph that combines two or more forms into one graph. In the example, the general guidelines for setting the width of the figure in the call to the *setPDF* function is 3.5 for a single column figure and 7.167 for a double-column figure; the maximum height is 9.333, all of the units are in inches. However, if a full page figure is desired, the layout can be set to “portrait” or “landscape” depending on the orientation needed by the figure. Comments in the code in figure 2 and in other code included in this report are preceded by the # symbol.

```
# Create the dataset to replicate the specific example of the
# combination graph. Setting WY to character sets up axes without ticks.
CG <- data.frame(WY=c("1998", "1999", "2000", "2001", "2002", "2003"),
  Prec=c(59.0, 33.7, 39.1, 47.1, 43.5, 69.8), stringsAsFactors=FALSE)
# Set up device for a 1/2 page width figure
setPDF(list(width=3.5, height=3), basename="fig2")
# Call to setLayout not needed as a single graph in the figure
# For the bar graph, set up a graph with nothing plotted.
# Note the use of as.expression to embed the en-dash in the x-axis
# caption. Expressions give the user a great deal of flexibility
# in formatting text for graphs.
AA.pl <- with(CG, xyPlot(WY, Prec, Plot=list(what="none"),
  ytitle="Precipitation, in inches", ylabel=5, yaxis.range=c(0, 80),
  xtitle=as.expression(substitute(0-S,
    list(0="Water year (October", S="September"))))))
# Add the bars. No explanation needed so no need to overwrite AA.pl
with(CG, addBars(WY, Prec, Bars=list(fill="lightblue"),
  current=AA.pl))
# The long-term average precipitation is 53.6 to add the line. Solid line
# used here--illustrator should convert to dashed if necessary.
# Accept default line color (black)
refLine(horizontal=53.6, current=AA.pl)
# Add the annotation. In this case, because the x-axis is discrete,
# the x-axis value for the middle must be computed from AA.pl; the x and
# y value must be numeric for addAnnotation.
# Note the \n in the annotation indicates a new line, producing stacked text.
addAnnotation(mean(AA.pl$xax$range), 53.6,
  "Long-term average\n53.6 inches", justification="center",
  current=AA.pl)
# Add the figure caption, generally completely redone by illustrator.
addCaption("Figure 2. Specific example of combination graph.")
# Finally close the current graphics device.
dev.off()
```

**Figure 2.** R code for a specific example of a combination graph.

## Programmer's Guide

This section of the report provides details about the structure of graphics set-up functions, main plotting functions, and functions that add to a graph. This section can be useful to anyone who wants to add a main plotting function or method to the package or develop a script that needs to call other plotting functions in base R. Many of the main plotting functions in **smwrGraphs** have methods for different types of arguments. The remainder of this section focuses on two functions that create or add to graphs, but the “HeatMap.r” demo script, found at <https://github.com/USGS-R/smwrGraphs/tree/master/demo>, also illustrates how to use low-level functions within the graphics environment in **smwrGraphs**.

### Device Set-Up Function

This section of the report uses the *setPDF* function to describe the necessary steps for setting up a graphics device to work with the functions in **smwrGraphs**. The arguments to the function must represent those needed by the device; in general, the size of the device and the filename are needed. Code used for examples in this report is included as boxed code as follows for the *setPDF* function:

```
setPDF <- function(layout="portrait", basename="USGS", multiplefiles=FALSE) {
```

The options for the line weight factor, the pdf file output, and the font size must be set. The line weight factor (*.lwt\_factor*) can generally be set to a specific value to control the line weights in the device, a value of 1 sets the correct values for line weights in a pdf file and can generally be used in any device. The options for pdf file output (*.pdf\_graph*) must be set to TRUE to guarantee that specialized actions required to produce editable graphics are performed by the low-level functions. The font size must be set. In this case, the font size is set to a variable (*fontSize*) used later when *pointsize* is set to *fontSize*.

```
## set global variables for linewidths and pdf
options(.lwt_factor = 1)
options(.pdf_graph = TRUE)
fontSize <- 8
```

The following boxed code sets up the size of the graphics device and the figure size in inches. The variables set are used later in the actual call to set up the device and setting up the initial graphics parameters in the call to *par*. Because the focus of the output from the plotting functions in **smwrGraphs** is the production of graphs for publications, a fixed size is necessary. Note that the *setRStudio* function (see appendix 1) is anomalous because this function permits resizing of graphs; however, some graphs cannot be created using that output device.

```
if(class(layout) == "list") { # custom
  width <- layout$wid
  height <- layout$hei
  fin <- c(width, height) - .1
}
else {
  layout=match.arg(layout, c("portrait", "landscape"))
  if(layout == "portrait") {
    fin <- c(7.25, 9.5)
    width=8.5
    height=11.
  }
  else { # landscape
    fin <- c(9.5, 7.25)
    width=11.
    height=8.5
  }
}
```

## 8 smwrGraphs—An R Package for Graphing Hydrologic Data, Version 1.1.2

The output file must be set up and created; an example of this is shown in the following boxed code. In most cases, width, height, and the font size will be necessary options to select.

```
name <- make.names(basename) # make a legal name
## set up the output
if(multiplefiles) # make name
  name <- paste(name, "%03d.pdf", sep="")
else
  name <- paste(name, ".pdf", sep="")
PDFFont <- "Helvetica-Narrow"
pdf(file=name, onefile=!multiplefiles, width=width, height=height,
    family=PDFFont, pointsize=fontSize, colormodel="cmyk", title=basename)
```

A font family called “USGS” must be set up for the device (as shown in the following boxed code) or warnings or errors will be generated from the main plotting functions and those functions that add any text. The actual function that registers the USGS font family will vary by device.

```
## set up for export to PDF file.
if(all(names(pdfFonts()) != "USGS")) # Check to see if already in the PDF font list
  pdfFonts("USGS" = Type1Font("Helvetica-Narrow",
    pdfFonts("HelveticaNarrow")[[1L]]$metrics))
```

The final section of code sets up any necessary graphical parameters as shown in the following boxed code. The only required parameters are the figure size and the style of the axis labels (`las`, which should be set to 1). For *invisible* function, the device number and basename of the output file are returned invisibly in a tagged list. The device number could be used in the call to *dev.off* to close the output file, if multiple graphics devices are opened.

```
dev <- dev.cur()
par(fin=fin, las=1)
invisible(list(dev=dev, name=basename))
}
```

## Main Plotting Function

This section of the report uses the *areaPlot* function to describe the main plotting function. Although the *areaPlot* function does not have methods for specific arguments, the extension to include different methods is straightforward and explained in many books and documentation for R. The example code is presented below in boxed code with comments.

The formal arguments are set up in consistent order in the following boxed code for this example. The first set of arguments are the data to plot, in this case, *x* and *y*. The next argument, *Areas*, in this example, describes how the data are to be plotted. The following set of arguments, beginning with *yaxis* or *xaxis*, set up the format of the axis. Not all axis format arguments may be appropriate for the graph; for example, *yaxis.rev*, which reverses the y-axis, is not supported for area plots. The remaining arguments set up the axis labels, axis titles (also called axis captions), and the figure caption. The last argument, *margin*, sets up specific margins, typically required only to force a specific size to the plot area. Additional plotting features, such as the reference or 1:1 line in the *qqPlot* function, typically follow the argument that describes how to plot the data.

```
areaPlot <- function(x, y, # data specs
  Areas=list(name="Auto", fillDir="between", base="Auto",
    lineColor="black", fillColors="pastel"), # Area controls
  yaxis.log=FALSE, yaxis.range=c(NA,NA), # y-axis controls
  xaxis.log=FALSE, xaxis.range=c(NA,NA), # x-axis controls
  ylabels=7, xlabels="Auto", # axis labels
  xtitle="",
  ytitle="", # axis titles, blank by default
  caption="", # caption
  margin=c(NA, NA , NA, NA)) { # margin control
```

The following boxed code sets up the information needed to properly display the data. The *setDefault*s function is used for arguments other than *Plot*, which used the *setPlot* function. The use of *setDefault*s is required to fill in default options not specified by the user. The call to *setGD* is a convenience call to set up the correct graphics environment if one has not already been set up. The remaining code sets up the x- and y-axis and the labeling. The functions in the *smwrGraphs* package that are used in this section include *setAxis*, which is typically called from *do.call*, and the functions to set up specific axis types, *datePretty*, *linearPretty*, *logPretty*, *probPretty*, *timePretty*, and *transPretty* (appendix 1). Other functions, like *isCharLike*, *isDateLike*, *isGroupLike*, or *isNumberlike* (in the *smwrBase* package (Lorenz, 2015)) can be used to help determine how to set up the axis, but would not necessarily be used in method functions because the type of data would be known. The *numericData* function forces all data to be in a consistent numeric format; for example, data of class “Date” or “POSIXct” would be converted to comparable numeric values.

```
##
## Set defaults for Areas
Areas <- setDefaults(areas, name="Auto", fillDir="between",
                    base="Auto", lineColor="black",
                    fillColor="pastel")
## Set up plot (either xy or time)
if(dev.cur() == 1)
  setGD("AreaPlot")
y <- as.matrix(y) # Force to matrix if data.frame, for example
xRng <- range(x, na.rm=TRUE)
yRng <- range(y, na.rm=TRUE)
## Adjust y range if needed
if(Areas$fillDir == "under" && Areas$base != "Auto")
  yRng[1] <- as.double(Areas$base)
if(is.list(ylabels))
  yax <- c(list(data=yRng, axis.range=yaxis.range, axis.log=yaxis.log,
              axis.rev=FALSE), ylabels)
else
  yax <- list(data=yRng, axis.range=yaxis.range, axis.log=yaxis.log,
            axis.rev=FALSE, axis.labels=ylabels)
yax <- do.call("setAxis", yax)
yax <- yax$dax
if(isDateLike(x)) { # Set up time axis
  if(any(is.na(xaxis.range)))
    tRng <- xRng
  else
    tRng <- xaxis.range
  xax <- datePretty(xaxis.range, major=xlabels)
  x <- numericData(x)
  xRng <- numericData(xRng)
}
else { # Regular x-axis
  if(!is.list(xlabels) && xlabels == "Auto")
    xlabels=7
  if(is.list(xlabels))
    xax <- c(list(data=xRng, axis.range=xaxis.range, axis.log=xaxis.log,
                axis.rev=FALSE), xlabels)
  else
    xax <- list(data=xRng, axis.range=xaxis.range, axis.log=xaxis.log,
              axis.rev=FALSE, axis.labels=xlabels)
  xax <- do.call("setAxis", xax)
  xax <- xax$dax
}
```



The *setMargin* function, as used in the following boxed code, replaces the NA values in the *margin* argument with values derived for the y-axis labeling. The x-axis labels are set to defaults that allow for typical labels and the graph title; see the *boxPlot* function (appendix 6) for ways to set up a wider x-axis margin, for example to accommodate rotated labels. The right-hand margin is set to a small value; thus, to allow for labels, the user must specify a value in *margin* in the call to *areaPlot* or other main plotting function. The remaining lines (except for the call to *par*) simply set up convenience arguments required later in the function.

```
## set margins and controls
margin.control <- setMargin(margin, yax)
margin <- margin.control$margin
right <- margin.control$right
top <- margin.control$top
left <- margin.control$left
bot <- margin.control$bot
par(mar=margin)
current=list(yaxis.log=yaxis.log, yaxis.rev=FALSE, xaxis.log=xaxis.log)
```

The call to *plot* in the following boxed code is needed to set up the graph. This call sets up the plotting environment for the subsequent plotting calls, but nothing is actually drawn on the graph. Nothing being plotted is very typical in the main plotting functions in **smwrGraphs** as it permits finer control of the graphical parameters, and thus, of what will be plotted.

```
##
plot(xRng, yRng, type='n', xlim=xax$range, xaxs='i', axes=FALSE,
     ylim=yax$range, yaxs='i', ylab="", xlab="")
```



The following boxed code actually draws the plot and is specific to the plotting function, and the general plotting structure will differ from plot type to plot type. The simplest example can be seen in the methods for the *xyPlot* function (appendix 7). Other main plotting functions may use base R functions to do the primary computations, but the plotting is done using the specific graphical parameters set up in *smwrGraphs*; see the *histGram* function (appendix 9) for an example of such a function. Examining code from a function that is similar to the desired new graph is a good way to proceed. The key to consistent line weights and symbology is to use the contents from the *setPlot* or *setDefault*s functions.

```
## Set up to add the areas
nAreas <- ncol(y)
if(Areas$fillDir == "between")
  nAreas <- nAreas - 1
if(Areas$name[1] == "Auto") { # Generate names
  if(Areas$fillDir == "between")
    Areas$name <- paste(colnames(y)[-1], colnames(y)[-ncol(y)], sep='-')
  else # Must be under
    Areas$name <- colnames(y)
}
if(length(Areas$fillColors) == 1 && nAreas > 1) { # Generate colors
  ColorGen <- get(paste(Areas$fillColors, "colors", sep=','))
  Areas$fillColors <- rev(ColorGen(nAreas)) # need to reverse because of order
}
plotted
}
else # Use what's there
  Areas$fillColors <- rev(setColor(Areas$fillColors))
## Add areas--needs work on the explanation
if(Areas$fillDir == "between") {
  for(i in seq(nAreas, 1)) {
    xToPlot <- c(x, rev(x))
    yToPlot <- c(y[, i+1L], rev(y[, i]))
    current <- addArea(xToPlot, yToPlot, Area=list(name=Areas$name[i],
                                                    color=Areas$fillColors[i],
                                                    outline=Areas$lineColor),
                      current=current)
  }
}
else { # Must be under
  if(Areas$base == "Auto")
    base <- yax$range[1L]
  else
    base <- Areas$base
  for(i in seq(nAreas, 1L)) {
    xToPlot <- c(x[1L], x, x[length(x)])
    yToPlot <- c(base, y[,i], base)
    current <- addArea(xToPlot, yToPlot, Area=list(name=Areas$name[i],
                                                    color=Areas$fillColors[i],
                                                    outline=Areas$lineColor),
                      current=current)
  }
}
}
```

## 12 smwrGraphs—An R Package for Graphing Hydrologic Data, Version 1.1.2

The following remaining boxed code simply finishes the graph, by drawing the bounding box, drawing ticks and labeling the axis, and setting up the object to return. The returned object must contain the axis information and information for the explanation. For *areaPlot*, the explanation information is added in the calls to *addArea*.

```
box(lwd=frameWt())
## label the axes
renderY(yax, lefttitle=ytile, left=left, right=right)
renderX(xax, bottitle=xtile, bottom=bot, top=top, caption=caption)
current$margin <- margin
## update x and y with the input
current$x <- x
current$y <- y
current$yax <- yax
current$xax <- xax
invisible(current)
}
```

### Function to Add to a Graph

The *addArea* function is used as an example in this section for adding a graph. The *addArea* function does not have methods for specific arguments, but the extension to include different methods is straightforward and explained in many books and documentation for R. The example code is presented below in boxed code with comments.

```
addArea <- function(x, y, ybase=NULL, # data
                  Area=list(name="", color="gray",
                            outline="black"), # area controls
                  current=list(yaxis.log=FALSE, yaxis.rev=FALSE,
                              xaxis.log=FALSE)) { # current plot parameters
```

In general, the list of arguments to functions that add to a graph are much shorter than the main plotting functions that create graphs. The first set of arguments in the preceding code box are the data to plot: *x*, *y*, and *ybase*. The next argument in this example is *Area*, but the argument *Plot* commonly is used; both of these arguments describe how the data are to be plotted. The last argument, *current*, typically contains the output from the main plotting function, but accepts the defaults if no axis was transformed and no explanation will be added.

The following boxed code sets up the data to plot. The *transData* or *numericData* function typically is used in this section to transform the data to match the axis transformations of the graph. For this example, the additional code forces the values in *ybase* to match the other data.

```
y <- transData(y, current$yaxis.log, current$yaxis.rev,
              current$ytrans, current$ytarg)
x <- transData(as.double(x), current$xaxis.log, FALSE,
              current$xtrans, current$xtarg)
N <- length(x)
if(!is.null(ybase)) { # Need to do something
  if(length(ybase) == 1L) { # a constant
    y <- c(ybase, y, ybase)
    x <- c(x[1L], x, x[N])
  }
  else { # ybase must be as long as y
    if(length(ybase) != N)
      stop("length of ybase must be one or match y")
    y <- c(y, rev(ybase))
    x <- c(x, rev(x))
  }
}
if(any(is.na(c(x, y))))
  stop("Missing value are not permitted in either x, y, or ybase")
```

The following boxed code sets the defaults for the *Area* argument, and demonstrates the *setPlot* and *setExplanation* functions, which are required to maintain the explanation information from each plot. For this example, the data are plotted using the *polygon* function.

```
## Set up plot information
Area <- setDefaults(Area, name="", color="gray", outline="black")
if(Area$outline == "none")
  Area$outline <- NA
Plot <- setPlot(list(), name=Area$name, what="points", type="solid",
               width="standard", symbol="none", filled=TRUE,
               size=0.09, color="black", area.color=Area$color,
               area.border=Area$outline) # force defaults if not set
explan <- setExplan(Plot, old=current$explanation) # add info to set up explanation
polygon(x, y, col=Area$color, border=Area$outline)
```

The final following boxed code simply updates the data that were plotted, updates the explanation, and returns that list. Updating the data and explanation is critical for subsequent functions that rely on that information.

```
current$x <- x
current$y <- y
current$explanation <- explan
invisible(current)
}
```

## Summary

This report describes an R package called **smwr-Graphs**, which consists of a collection of graphing functions for hydrologic data within R, a programming language and software environment for statistical computing. The functions in the package have been developed by the U.S. Geological Survey to create high-quality graphs for publication or presentation of hydrologic data that meet U.S. Geological Survey graphics guidelines.

In addition to describing the functions that are within **smwrGraphs**, a Programmer's Guide section that is intended to help programmers contribute new functions to the package is included. The Programmer's Guide section breaks down one main plotting function and one function that adds to a graph into sections and describes the general approach and specific low-level functions used in that section.

## Disclaimer

This package was written by U.S. Federal government employees in the course of their employment and is therefore in the public domain, which means it is not copyrighted and use is unlimited; however, some of the functions depend on other R-packages, which, although free and open source, have more restrictive licensing. Those packages are digest and lubridate [GNU (Gnu's Not Unix) GPL (General Public License), memoise (Massachusetts Institute of Technology,

MIT), XML (Berkeley Software Distribution, BSD). R itself is released under the free software license GNU GPL, either Version 2, June 1991, or Version 3, June 2007. Additional information on licensing is available at <https://www.r-project.org/Licenses/> and <https://www.gnu.org/licenses/license-list.html#SoftwareLicenses>.

Although this software package has been used by the U.S. Geological Survey (USGS), no warranty, expressed or implied, is made by the USGS or the U.S. Government as to the accuracy and functioning of the program and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith. This software and related material (functions and documentation) are made available by the USGS to be used in the public interest and the advancement of science. Users may, without any fee or cost, use, copy, modify, or distribute this software, and any derivative works thereof, and its supporting documentation, subject to the USGS Software User's Rights Notice, <https://water.usgs.gov/software/help/notice/>.

## Acknowledgments

The authors thank the time and effort of the U.S. Geological Survey personnel who have used and reviewed this package and contributed suggestions to improve the package. The reviewers were Jared Trost, Minnesota Water Science Center; and Laura De Cicco, Center for Integrated Data

Analytics. The publication personnel Rebecca Inman, Katharine Laub, Sue Roberts, and Caryl Wipperfurth contributed suggestions and feedback to make the output from the *setPDF* function quick and easy for USGS illustrators to prepare graphs for USGS publication.

## References Cited

- Helsel, D.R., and Hirsch, R.M., 2002, Statistical methods in water resources: U.S. Geological Survey Techniques of Water Resources Investigations, book 4, chapter A3, 522 pages. (Also available at <http://pubs.usgs.gov/twri/twri4a3/>).
- Lorenz, D.L., 2015, smwrBase—An R package for managing hydrologic data, version 1.1.1: U.S. Geological Survey Open-File Report 2015–1202, 7 p. [Also available at <http://dx.doi.org/10.3133/ofr20151202>.]
- Lorenz, D.L., Ahearn, E.A., Carter, J.M., Cohn, T.A., Dan-chuk, W.J., Frey, J.W., Helsel, D.R., Lee, K.E., Leeth, D.C., Martin, J.D., McGuire, V.L., Neitzert, K.M., Robertson, D.M., Slack, J.R., Starn, J., Vecchia, A.V., Wilkison, D.H., and Williamson, J.E., 2011, USGS library for S-PLUS for Windows—Release 4.0: U.S. Geological Survey Open-File Report 2011–1130. [Also available at <http://pubs.er.usgs.gov/publication/ofr20111130>.]
- Piper, A.M., 1944, A graphic procedure in the geochemical interpretation of water analyses: American Geophysical Union Transactions, v. 25, p. 914–923.
- R Development Core Team, 2011, R Installation and Administration, Version 2.14.1, 2011-12-22: accessed January 13, 2012, at <http://streaming.stat.iastate.edu/CRAN/doc/manuals/R-admin.pdf>.
- Stiff, H.A., Jr., 1951, The interpretation of chemical water analysis by means of patterns: Journal of Petroleum Technology, v. 3, no. 10, p. 15–17.
- Venables, W.N., Smith, D.M., and the R Development Core Team, 2011, An Introduction to R, Version 2.13.1, 2011-07-08: accessed August 11, 2011, at <http://cran.r-project.org/doc/manuals/R-intro.pdf>.

## Appendixes

---

## **Appendix 1. R Documentation**

The R documentation is available as a Portable Document Format (PDF) file and can be accessed at <https://doi.org/10.3133/ofr20161188>.

## **Appendix 2. Graph Setup Vignette**

Vignettes are the established R community method for providing examples of how to use the package. The vignette for graph setup is available as a Portable Document Format (PDF) file that describes setting up the graphics environment for **smwrGraphs** functions and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples covered in this vignette include simple graphs, plot customization, axis customization, and multiple graphs.

## **Appendix 3. Graph Additions Vignette**

Vignettes are the established R community method for providing examples of how to use the package. The vignette for graph additions is available as a Portable Document Format (PDF) file that describes adding plots and other features to graphs created by **smwrGraphs** functions and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples covered in this vignette include reference lines with annotation, grid, smoothed, and regression lines.

## **Appendix 4. Date Axis Formats Vignette**

Vignettes are the established R community method for providing examples of how to use the package. The vignette for date axis formats is available as a Portable Document Format (PDF) file that describes setting up date and time axis formats for **smwrGraphs** graphs and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples covered in this vignette include hour, day, month, year, and water year formats.

## **Appendix 5. Graph Gallery Vignette**

Vignettes are the established R community method for providing examples of how to use the package. The vignette for the graph gallery is available as a Portable Document Format (PDF) file that describes miscellaneous **smwrGraphs** graphs not covered in other vignettes, such as Stiff diagrams (Stiff, 1951), and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples covered in this vignette include biplots, scaleless graphs, area graphs, contour plots, correlograms, surface plots, and dendograms.

## **Appendix 6. Boxplot Vignette**

Vignettes are the established R community method for providing examples of how to use the package. The vignette boxplots is available as a Portable Document Format (PDF) file that describes how to create boxplots using **smwrGraphs** functions and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples in this vignette include various boxplot types.

## **Appendix 7. Line and Scatter Vignette**

Vignettes are the established R community method for providing examples of how to use the package. The vignette for line and scatter plots is available as a Portable Document Format (PDF) file that describes how to create line and scatter plots using **smwrGraphs** functions and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples in this vignette include scatter, date/time, season, series, color, dot, and scaled plots.

## Appendix 8. Piper Plot Vignette

Vignettes are the established R community method for providing examples of how to use the package. The vignette for Piper plots (Piper, 1944) is available as a Portable Document Format (PDF) file that describes how to create Piper plots using **smwrGraphs** functions and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples in this vignette include Piper plots, ternary diagrams, and Stiff diagrams.

## Appendix 9. Probability Plots Vignette

Vignettes are the established R community method for providing examples of how to use the package. The vignette for probability plots is available as a Portable Document Format (PDF) file that describes how to create probability plots using **smwrGraphs** functions and can be accessed at <https://doi.org/10.3133/ofr20161188>. Examples in this vignette include probability plots, Q-normal plots, Q-Q plots, and histograms.

**Publishing support provided by:**

Rolla and Pembroke Publishing Service Centers

**For more information concerning this publication, contact:**

Director, USGS Minnesota Water Science Center  
2280 Woodale Drive  
Mounds View, Minnesota 55112  
(763) 783-3100

**Or visit the Minnesota Water Science Center website at:**

<https://mn.water.usgs.gov/>

